Phần 1: Mô tả hệ thống

- Actor khách hàng:
    - Đăng ký, đăng nhập, đăng xuất tài khoản người dùng
    - Thay đổi thông tin tài khoản
    - Truy cập vào danh sách các xe trên hệ thống
    - Tìm kiếm xe
    - Xem thông tin xe, hãng xe
    - Đặt yêu cầu thuê xe, chỉnh sửa thông tin yêu cầu thuê
    - Xem danh sách các yêu cầu thuê, các hợp đồng thuê của tài khoản
- Actor quản lý:
    - Đăng ký, đăng nhập, đăng xuất tài khoản quản lý
    - Thay đổi thông tin tài khoản quản lý
    - Quản lý xe:
        + Thêm xe
        + Sửa thông tin xe
        + Xoá xe
        + Xem thông tin xe
    - Quản lý hãng xe:
        + Thêm hãng xe
        + Sửa thông tin hãng xe
        + Xoá hãng xe
        + Xem thông tin hãng xe
    - Quản lý yêu cầu thuê:
        + Xoá yêu cầu thuê
        + Xem thông tin yêu cầu thuê
    - Quản lý hợp đồng thuê:
        + Thêm hợp đồng thuê
        + Sửa thông tin hợp đồng thuê
        + Xoá hợp đồng thuê
        + Xem thông tin hợp đồng thuê
    - Quản lý khách hàng:
        + Thêm khách hàng
        + Sửa thông tin khách hàng
        + Xoá khách hàng
        + Xem thông tin khách hàng

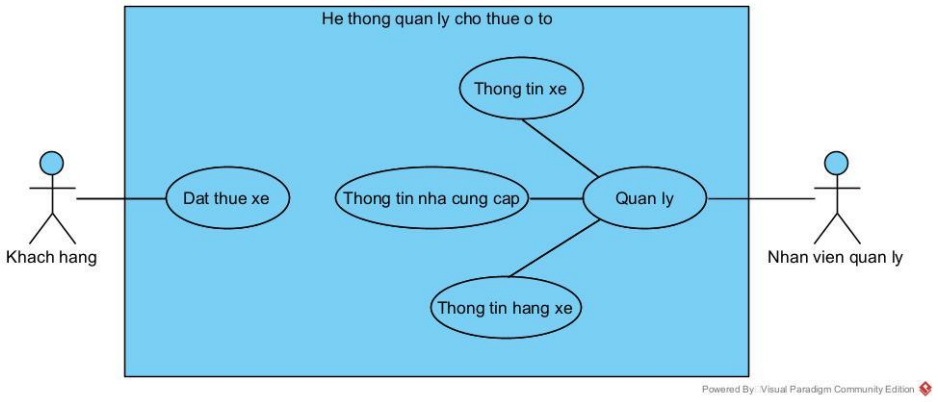
Actor	Hành động	Mô tả chi tiết
Người dùng	Đăng ký tài khoản	

	Đăng nhập tài khoản	Đăng nhập bằng username và mật khẩu
	Đăng xuất tài khoản	
	Thay đổi thông tin tài khoản	
	Truy cập vào danh sách các xe trên hệ thống	Truy cập thông qua trang chủ, trang tìm kiếm,...
	Tìm kiếm xe	Tìm kiếm xe tại trang tìm kiếm sử dụng từ khoá, lựa chọn hãng xe, năm sản xuất,...
	Đặt yêu cầu thuê xe	
	Chỉnh sửa thông tin yêu cầu thuê	
	Xem danh sách các yêu cầu thuê	
	Xem danh sách các hợp đồng thuê	
Quản lý	Đăng ký tài khoản	
	Đăng nhập tài khoản	Đăng nhập bằng username và mật khẩu
	Đăng xuất tài khoản	
	Thay đổi thông tin tài khoản	
	Thêm xe	
	Sửa thông tin xe	
	Xoá xe	
	Xem thông tin xe	
	Thêm hãng xe	
	Sửa thông tin hãng xe	

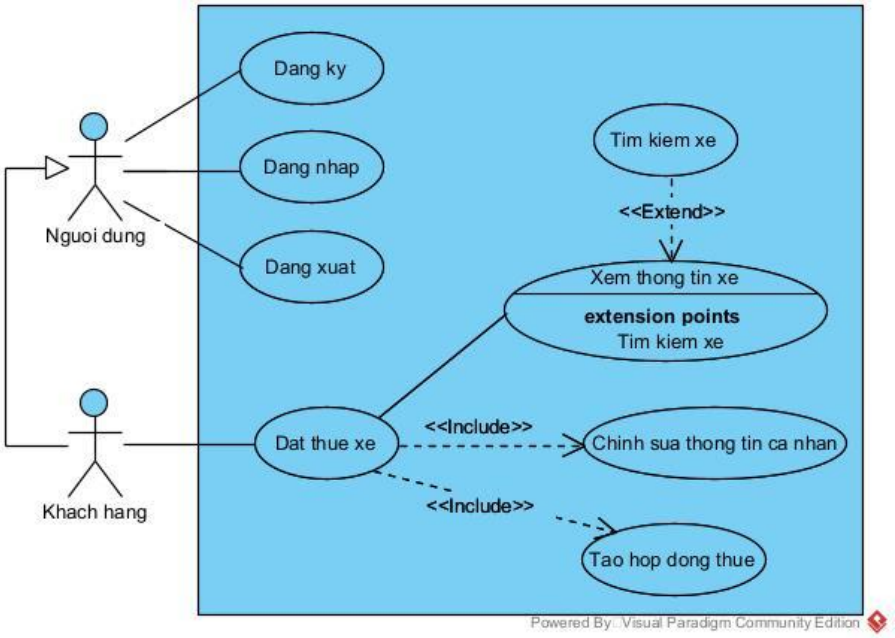
	Xoá hãng xe	
	Xem thông tin hãng xe	
	Xoá yêu cầu thuê	
	Xem thông tin yêu cầu thuê	
	Thêm hợp đồng thuê	
	Sửa thông tin hợp đồng thuê	
	Xoá hợp đồng thuê	
	Xem thông tin hợp đồng thuê	
	Thêm khách hàng	
	Sửa thông tin khách hàng	
	Xoá khách hàng	
	Xem thông tin khách hàng	

Phần 2: Biểu đồ use case

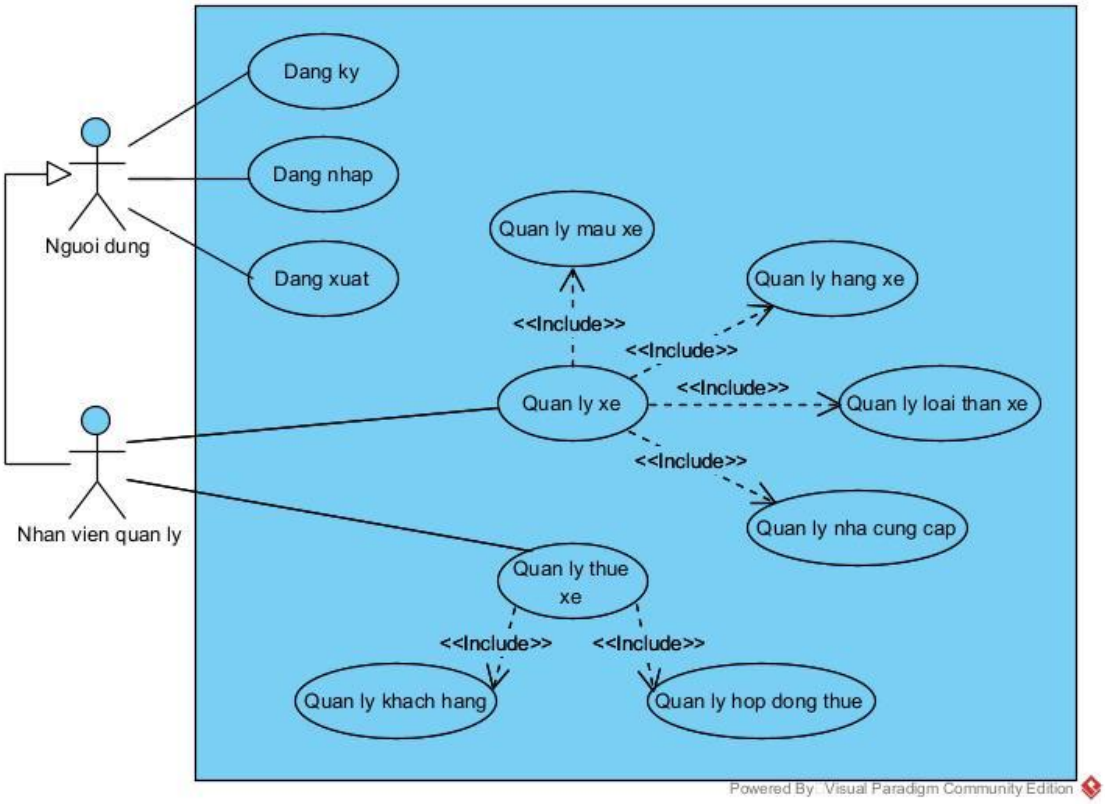
- Use case tổng quát:



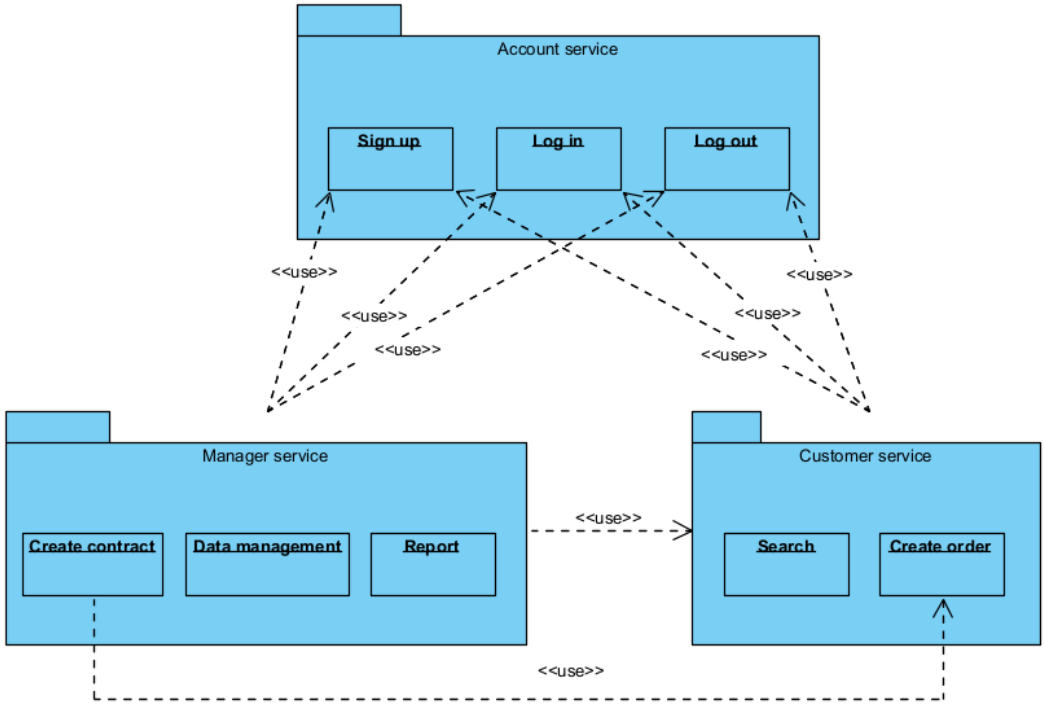
- Use case khách hàng:



- Use case quản lý:



Phần 3: Biểu đồ phân rã



Phần 4: Dạng communication

Giao tiếp đồng bộ:

```
import requests


def synchronous_communication():
        # Gửi yêu cầu đồng bộ
        response = requests.get("https://api.service1.com/data")


        # Kiểm tra mã trạng thái
        if response.status_code == 200:
```

```
    data = response.json()
    # Xử lý dữ liệu

synchronous_communication()
```

Giao tiếp bất đồng bộ:

```
import requests
import asyncio

async def asynchronous_communication():
    loop = asyncio.get_event_loop()
    # Gửi yêu cầu bất đồng bộ
    response = await loop.run_in_executor(None, requests.get,
"https://api.service2.com/data")

    # Kiểm tra mã trạng thái
    if response.status_code == 200:
    data = response.json()
    # Xử lý dữ liệu

asyncio.run(asynchronous_communication())
```

a. Ưu điểm:

- Giao tiếp không đồng bộ làm giảm khớp nối giữa các dịch vụ
- Có thể xử lý khối lượng lớn tin nhắn và hỗ trợ cân bằng tải
- Hỗ trợ các mẫu nhắn tin khác nhau như xuất bản-đăng ký, điểm-điểm và yêu cầu-trả lời

b. Nhược điểm:

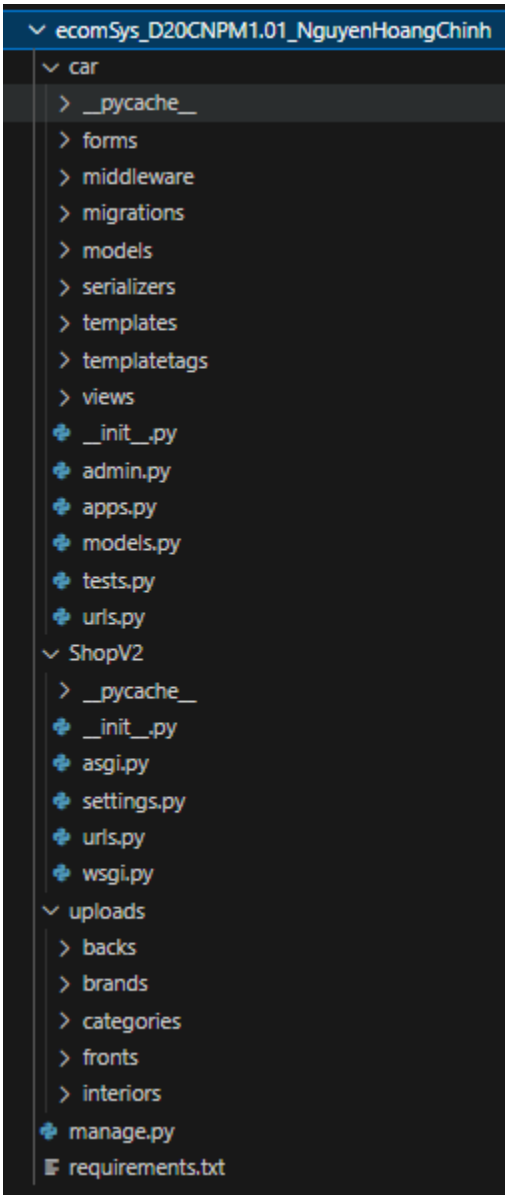
- Giới thiệu sự phức tạp do các thành phần bổ sung (môi giới tin nhắn)

- Có thể là thách thức để theo dõi luồng tin nhắn trong một hệ thống phân tán

- Độ trễ bổ sung so với giao tiếp trực tiếp

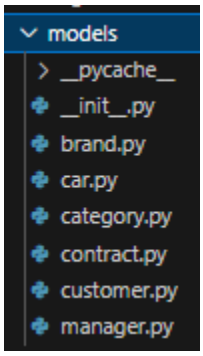
PHẦN 1: Hệ thống ban đầu

- Bài tập được thực hiện qua việc thiết kế lại một hệ thống cho thuê xe ô tô đơn giản, cho phép người dùng tìm kiếm, xem thông tin xe, đăng nhập, đăng ký tài khoản, đặt hợp đồng thuê.
- Kiến trúc ban đầu của project:



Hình 1

- Tệp models, bao gồm các model tương ứng với các bảng trong CSDL MySQL:



Hình 2

```python
from django.db import models

class Brand(models.Model):
    name= models.CharField(max_length=50)
    website= models.CharField(max_length=50)
    desintext= models.CharField(max_length=250, default='', blank=True, null= True)
    image= models.ImageField(upload_to='uploads/brands/')

    @staticmethod
    def get_all_categories():
        return Brand.objects.all()

    def __str__(self):
        return self.name

    class Meta:
        db_table = 'brand'
```

Hình 3: model của hãng xe Brand

```python
from django.db import models
from .category import Category
from .brand import Brand

class Car(models.Model):
    name= models.CharField(max_length=50)
    price= models.IntegerField(default=0)
    model= models.CharField(max_length=50,null=True)
    brand= models.ForeignKey(Brand,on_delete=models.CASCADE,default=1 )
    category= models.ForeignKey(Category,on_delete=models.CASCADE,default=1 )
    year= models.IntegerField(default=0)
    desintext= models.CharField(max_length=250, default='', blank=True, null= True)
    front= models.ImageField(upload_to='uploads/fronts/')
    back= models.ImageField(upload_to='uploads/backs/')
    interior= models.ImageField(upload_to='uploads/interiors/')
    instock= models.IntegerField(default=0)

    @staticmethod
    def get_cars_by_id(ids):
        return Car.objects.filter (id__in=ids)
    @staticmethod
    def get_all_cars():
        return Car.objects.all()

    @staticmethod
    def get_all_cars_by_categoryid(category_id):
        if category_id:
            return Car.objects.filter (category=category_id)
        else:
            return Car.get_all_cars()

    class Meta:
        db_table = 'car'
```

Hình 4: model của xe Car

```python
from django.db import models

class Category(models.Model):
    name= models.CharField(max_length=50)
    image= models.ImageField(upload_to='uploads/categories/')

    @staticmethod
    def get_all_categories():
        return Category.objects.all()

    def __str__(self):
        return self.name

    class Meta:
        db_table = 'category'
```

Hình 5: model của loại xe Category

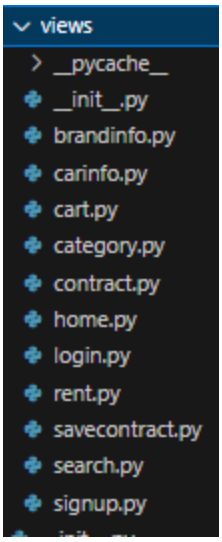
```python
from django.db import models
from django.forms import fields
from .car import Car
from .brand import Brand
from .customer import Customer
import datetime
from datetime import date

class Contract(models.Model):
    car = models.ForeignKey(Car,
                            on_delete=models.CASCADE)
    customer = models.ForeignKey(Customer,
                             on_delete=models.CASCADE)
    quantity = models.IntegerField(default=1)
    price = models.PositiveBigIntegerField()
    date = models.DateField (default=datetime.datetime.today)
    end_date = models.DateField(default=datetime.datetime.today)


    def placeOrder(self):
        self.save()

    @staticmethod
    def get_contracts_by_customer(customer_id):
        return Contract.objects.filter(customer=customer_id)

    @property
    def is_past_due(self):
        return date.today() <= self.end_date

    @property
    def car_name(self):
        car = Car.objects.get(id=self.car)
        brand = Brand.objects.get(id=car.brand)
        return brand.name +" "+ car.model +" "+ str(car.year)

    def __str__(self):
        return self.customer.first_name +" "+ self.customer.last_name +", from "+ str(self.date) +" to "+ str(self.end_date)

    class Meta:
        db_table = 'contract'
```

Hình 6: model của hợp đồng thuê Contract

```python
from django.db import models

class Customer(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField (max_length=50)
    phone = models.CharField(max_length=10)
    email=models.EmailField()
    password = models.CharField(max_length=100)

    #to save the data
    def register(self):
        self.save()


    @staticmethod
    def get_customer_by_email(email):
        try:
            return Customer.objects.get(email= email)
        except:
            return False

    @staticmethod
    def get_customer_by_id(customer_id):
        return Customer.objects.filter(id=customer_id)


    def isExists(self):
        if Customer.objects.filter(email = self.email):
            return True

        return False

    def __str__(self):
        return self.first_name +" "+ self.last_name

    class Meta:
        db_table = 'customer'
```

Hình 7: model của người dùng Customer

- Tệp views, bao gồm các hàm lấy dữ liệu từ CSDL và trả về, tương ứng với các bảng trong CSDL và một số chức năng:



Hình 8

```python
from django.shortcuts import render, redirect
from car.models.brand import Brand
from car.models.car import Car

# Create your views here.

def get_brand(request,pk):
    brand = Brand.objects.filter(id=pk)
    car = Car.objects.select_related("brand").filter(brand=pk)
    context = {'brand': brand, 'car': car}
    return render(request,'brandinfo.html',context)
```

Hình 9: hàm get_brand() lấy và trả về thông tin một hãng xe và danh sách các xe thuộc hãng trong brandinfo.py, chuyển đến brandinfo.html

```python
from django.shortcuts import render, redirect, HttpResponseRedirect
from car.models.car import Car
from car.models.category import Category
from django.views import View

def get_car(request,pk):

    carinfo = Car.objects.select_related('brand').filter(id=pk)
    for c in carinfo:
        carbrand = Car.objects.select_related('brand').filter(brand=c.brand)[:10]
        carcategory = Car.objects.select_related('brand').filter(category=c.category)[:10]
    context = {'carinfo': carinfo,'carbrand': carbrand,'carcategory': carcategory}
    return render(request,'carinfo.html',context)
```

Hình 10: Hàm get_car() lấy và trả về thông tin một xe và danh sách các xe cùng hãng và loại xe trong carinfo.py, chuyển đến carinfo.html

```python
from django.shortcuts import render, redirect, HttpResponseRedirect
from car.models.car import Car
from car.models.category import Category
from django.views import View

def category(request,pk):
    category = Category.objects.all()
    car = Car.objects.select_related('brand').all()
    if pk != 0:
        car = car.filter(category=pk)

    context = {'category': category,'car': car}
    return render(request,'category.html', context)
```

Hình 11: hàm category() lấy và trả về danh sách các loại xe và danh sách các xe thuộc một loại xe được chọn trong category.py, chuyển đến category.html

```python
from django.shortcuts import render, redirect
from django.contrib.auth.hashers import check_password
from car.models.customer import Customer
from django.views import View
from car.models.car import Car
from car.models.contract import Contract
from car.models.customer import Customer
from car.middleware.auth import auth_middleware
from django.db import connection

class ContractView(View):


    def get(self , request ):
        customer = request.session.get('customer')

        dict = {'customer': customer}
        contracts = Contract.objects.raw('''SELECT contract.*, car.model AS cmodel, car.year AS cyear, brand.name AS bname
                                            FROM contract
                                            INNER JOIN car
                                            ON contract.car_id = car.id
                                            INNER JOIN brand
                                            ON car.brand_id = brand.id
                                            WHERE contract.customer_id = %(customer)s
                                            ORDER BY contract.end_date DESC''', dict)

        customerinfo = Customer.get_customer_by_id(customer)
        return render(request , 'account.html'  , {'contracts' : contracts,'customerinfo': customerinfo})
```

Hình 12: lớp ContractView() lấy và trả về thông tin của tài khoản người dùng đang đăng nhập và danh sách các hợp đồng thuê của người dùng đó trong contract.py, chuyển đến account.html

```python
from django.shortcuts import render, redirect
from car.models.car import Car
from car.models.category import Category
from car.models.brand import Brand

def homepage(request):

    cart = request.session.get('cart')
    if not cart:
        request.session['cart'] = {}

    carTop3 = Car.objects.raw("SELECT * FROM car ORDER BY RAND() LIMIT 3")
    carMostPopular = Car.objects.select_related('brand').all()[:10]
    brandMostPopular = Brand.objects.raw("SELECT * FROM brand ORDER BY RAND() LIMIT 10")
    carCategory = Category.objects.raw("SELECT * FROM category ORDER BY RAND()")
    context = {'cartopthree': carTop3,'carMostPopular': carMostPopular,'brandMostPopular': brandMostPopular,'carCategory': carCategory}
    return render(request,'homepage.html',context)
```

Hình 13: hàm homepage() lấy và trả về danh sách các xe, hãng xe và loại xe theo tiêu chí nhất định trong home.py, chuyển đến homepage.html

```python
from django.shortcuts import render , redirect , HttpResponseRedirect
from django.contrib.auth.hashers import  check_password
from car.models.customer import Customer
from django.views import View


class Login(View):
    return_url = None

    def get(self, request):
        Login.return_url = request.GET.get ('return_url')
        return render (request, 'login.html')

    def post(self, request):
        email = request.POST.get ('email')
        password = request.POST.get ('password')
        customer = Customer.get_customer_by_email (email)
        error_message = None
        if customer:
            flag = check_password (password, customer.password)
            if flag:
                request.session['customer'] = customer.id

                if Login.return_url:
                    return HttpResponseRedirect (Login.return_url)
                else:
                    Login.return_url = None
                    return redirect ('homepage')
            else:
                error_message = 'Invalid !!'
        else:
            error_message = 'Invalid !!'

        print (email, password)
        return render (request, 'login.html', {'error': error_message})

def logout(request):
    request.session.clear()
    return redirect('login')
```

Hình 14: lớp Login() xác nhận đăng nhập, đăng xuất tài khoản người dùng trong login.py, với thông tin đăng nhập nhận từ login.html

```python
from django.shortcuts import render, redirect

from django.contrib.auth.hashers import check_password
from car.models.customer import Customer
from django.views import View

from car.models.car import Car
from car.models.contract import Contract

from car.forms.contract import ContractForm

import datetime

import json

class ContractSave(View):
    def post(self, request):
        postData = request.POST
        car_in = postData.get('car')
        customer_in = request.session.get('customer')
        quantity_in = postData.get('quantity')
        price = (Car.objects.get(id=car_in)).price
        date_in = datetime.datetime.now()
        end_date_in = postData.get('enddate')

        value = {
            'car': car_in,
            'quantity': quantity_in,
            'date': date_in,
            'end_date': end_date_in
        }
        error_message = None

        contract = Contract(car=Car.objects.get(id=car_in),customer=Customer.objects.get(id=customer_in),quantity=quantity_in,price=price,date=date_in,end_date=end_date_in)

        if int((datetime.datetime.strptime(end_date_in, '%Y-%m-%d')-date_in).days) < 1:
            contract.price = (Car.objects.get(id=car_in)).price * 1 * int(contract.quantity)
        else:
            contract.price = (Car.objects.get(id=car_in)).price * (int((datetime.datetime.strptime(end_date_in, '%Y-%m-%d')-date_in).days)+1) * int(contract.quantity)
        print(contract.price)

        contract.placeOrder()

        return redirect ('account')
```

Hinh 15: lớp ContractSave() nhập và lưu thông hợp đồng mới được tạo trong savecontract.py, với thông tin hợp đồng lấy từ contract.html

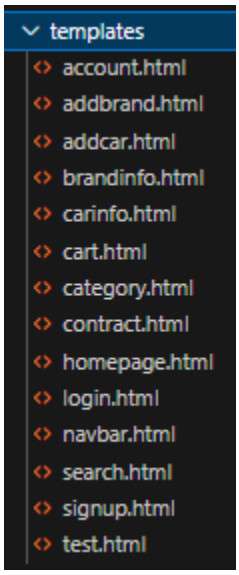
```python
from django.shortcuts import render, redirect, HttpResponseRedirect
from car.models.car import Car
from car.models.brand import Brand
from car.models.category import Category
from django.views import View

def search(request):
    model_in = request.POST.get('model')
    brand_in = request.POST.get('brand')
    category_in = request.POST.get('category')
    year_in = request.POST.get('year')

    carsearch = Car.objects.select_related('brand').all()
    brand_opt = Brand.objects.all()
    category_opt = Category.objects.all()
    if model_in != "All":
        carsearch = carsearch.filter(model=model_in)
    if brand_in != "0":
        carsearch = carsearch.filter(brand=brand_in)
    if category_in != "0":
        carsearch = carsearch.filter(category=category_in)
    if year_in != "All":
        carsearch = carsearch.filter(year=year_in)

    context = {'carsearch': carsearch,'brand_opt': brand_opt,'category_opt': category_opt}
    return render(request,'search.html',context)
```

Hình 16: hàm search() lấy và trả về danh sách kết quả tìm kiếm trong search.py, với thông tin tìm kiếm nhận từ search.html

```python
from django.shortcuts import render, redirect
from django.contrib.auth.hashers import make_password
from car.models.customer import Customer
from django.views import View


class Signup (View):
    def get(self, request):
        return render (request, 'signup.html')

    def post(self, request):
        postData = request.POST
        first_name = postData.get ('firstname')
        last_name = postData.get ('lastname')
        phone = postData.get ('phone')
        email = postData.get ('email')
        password = postData.get ('password')
        # validation
        value = {
            'first_name': first_name,
            'last_name': last_name,
            'phone': phone,
            'email': email
        }
        error_message = None

        customer = Customer (first_name=first_name,
                             last_name=last_name,
                             phone=phone,
                             email=email,
                             password=password)
        error_message = self.validateCustomer (customer)

        if not error_message:
            print (first_name, last_name, phone, email, password)
            customer.password = make_password (customer.password)
            customer.register ()
            return redirect ('homepage')
        else:
            data = {
                'error': error_message,
                'values': value
            }
            return render (request, 'signup.html', data)

    def validateCustomer(self, customer):
        error_message = None
        if (not customer.first_name):
            error_message = "Please Enter your First Name !!"
        elif len (customer.first_name) < 3:
            error_message = 'First Name must be 3 char long or more'
        elif not customer.last_name:
            error_message = 'Please Enter your Last Name'
        elif len (customer.last_name) < 3:
            error_message = 'Last Name must be 3 char long or more'
        elif not customer.phone:
            error_message = 'Enter your Phone Number'
        elif len (customer.phone) < 10:
            error_message = 'Phone Number must be 10 char Long'
        elif len (customer.password) < 5:
            error_message = 'Password must be 5 char long'
        elif len (customer.email) < 5:
            error_message = 'Email must be 5 char long'
        elif customer.isExists ():
            error_message = 'Email Address Already Registered..'
        # saving

        return error_message
```

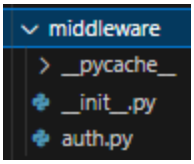
Hình 17: lớp Signup() xác nhận thông tin tạo tài khoản người dùng mới trong signup.py, với thông tin nhận từ signup.html

- Tệp templates, bao gồm các trang HTML có nhiệm vụ hiển thị các dữ liệu được các hàm trong tệp views trả về, tương ứng với các chức năng nhất định:



Hình 18

- Tệp middleware, bao gồm hàm xác định trạng thái đăng nhập, đăng xuất tài khoản của người dùng (Customer) sử dụng session:



Hình 19

```python
from django.shortcuts import redirect

def auth_middleware(get_response):
    # One-time configuration and initialization.

    def middleware(request):
        print(request.session.get('customer'))
        returnUrl = request.META['PATH_INFO']
        print(request.META['PATH_INFO'])
        if not request.session.get('customer'):
            return redirect(f'login?return_url={returnUrl}')

        response = get_response(request)
        return response

    return middleware
```

Hình 20: hàm auth_middleware() xác nhận trạng thái đăng nhập, đăng xuất của tài khoản người dùng trong auth.py

- Khai báo CSDL dược sử dụng trong settings.py của tệp ShopV2:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'shopv2',
        'USER':'root',
        'PASSWORD':'1432000',
        'HOST':'localhost',
        'PORT':'3306',
    }
}
```

Hình 21

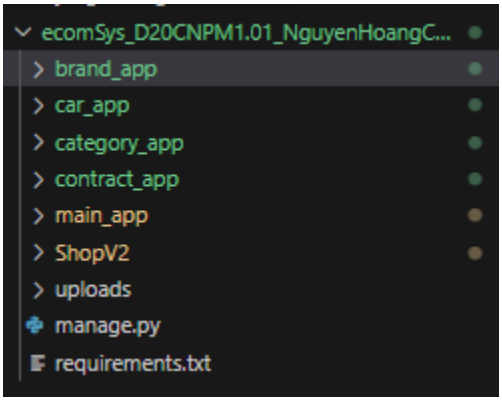
- Khai báo các URL trong urls.py của tệp car:

```python
from django.contrib import admin
from django.urls import path
from .views.home import homepage
from .views.signup import Signup
from .views.login import Login , logout
from .views.cart import Cart
from .views.rent import create_contract
from .views.savecontract import ContractSave
from .views.contract import ContractView
from .views.carinfo import get_car
from .views.brandinfo import get_brand
from .views.search import search
from .views.category import category
from .middleware.auth import  auth_middleware


urlpatterns = [
    path('', homepage, name='homepage'),

    path('homepage', homepage, name='homepage'),

    path('signup', Signup.as_view(), name='signup'),

    path('login', Login.as_view(), name='login'),

    path('logout', logout , name='logout'),


    path('search', search , name='search'),

    path('category/<int:pk>', category , name='category'),

    path('carinfo/<int:pk>', get_car, name='get-car'),

    path('brandinfo/<int:pk>', get_brand, name='get-brand'),

    path('rentcar', auth_middleware(create_contract), name='rent-car'),

    path('processcontract', auth_middleware(ContractSave.as_view()), name='process-contract'),

    path('account', auth_middleware(ContractView.as_view()), name='account'),
]
```

Hình 22

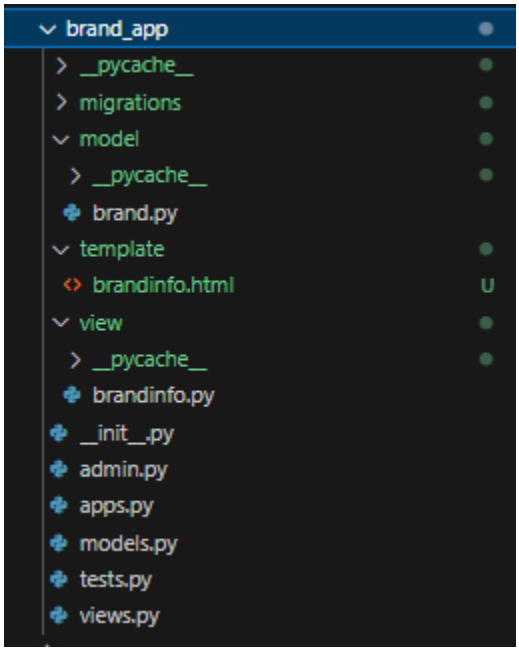
PHẦN 2: Chuyển đổi sang thiết kế microservices:

- Mục tiêu: tách hệ thống ban đầu thành các app nhỏ hơn, phụ trách một số chức năng hoặc một bảng trong CSDL.
- Kiến trúc mới của hệ thống:



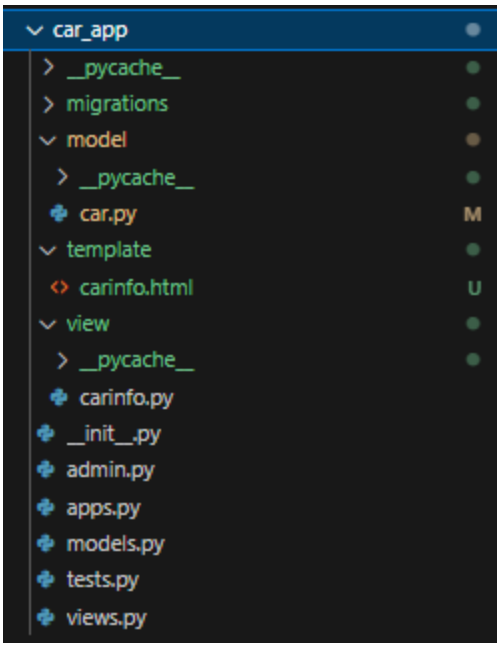
Hình 23

- App brand_app phụ trách dữ liệu hãng xe:



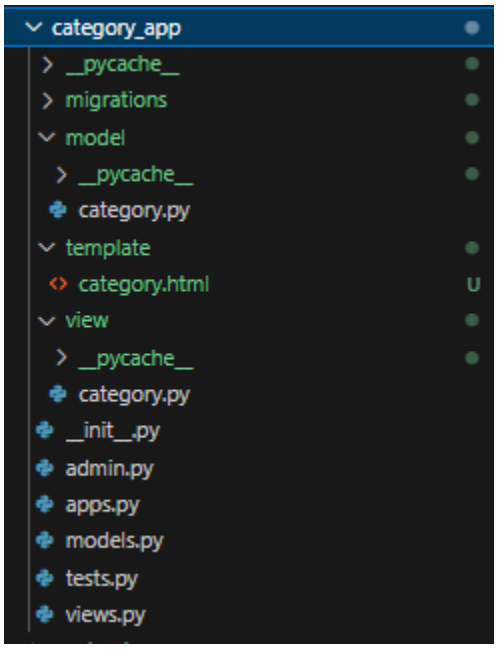
Hình 24

- App car_app phụ trách dữ liệu xe:



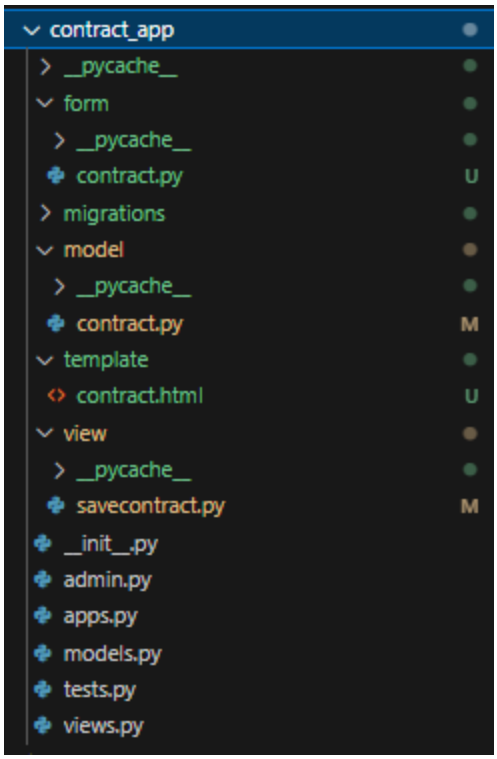
Hình 25

- App category_app phụ trách dữ liệu loại xe:



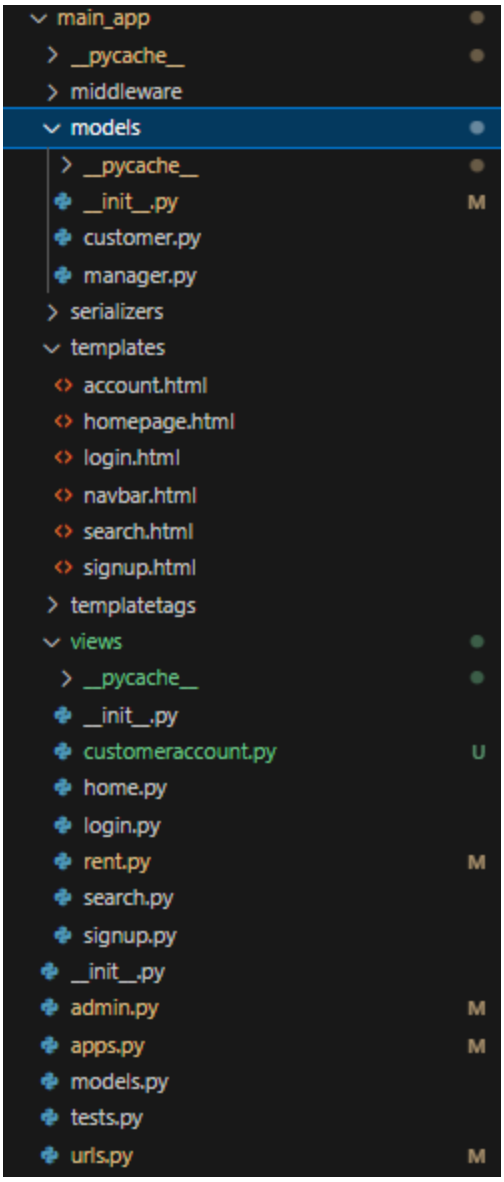
Hình 26

- App contract_app phụ trách dữ liệu hợp đồng thuê:



Hình 27

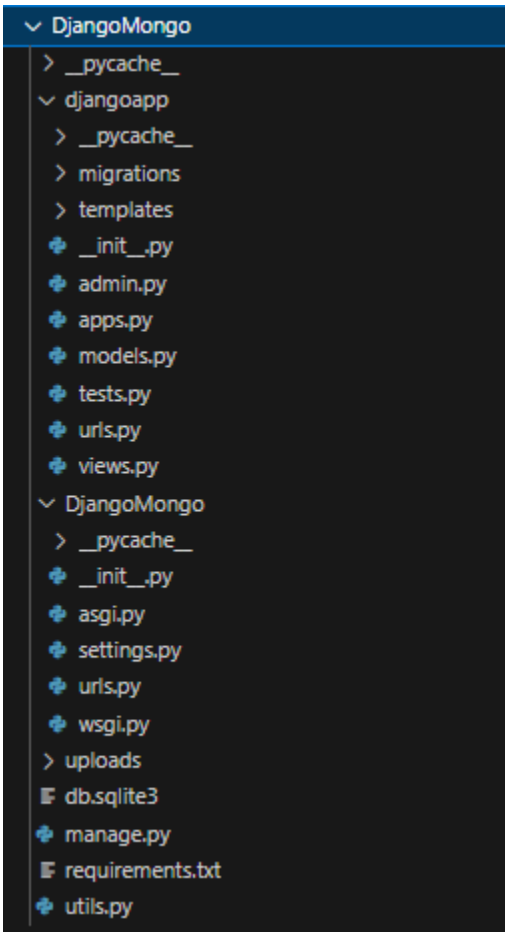
- App chính, phụ trách dữ liệu tài khoản người dùng, với urls.py bao gồm toàn bộ các url của hệ thống:


main_app
__pycache__
middleware
models
__pycache__
__init__.py
M
customer.py
manager.py
serializers
templates
account.html
homepage.html
login.html
navbar.html
search.html
signup.html
templatetags
views
__pycache__
__init__.py
customeraccount.py
U
home.py
login.py
rent.py
M
search.py
signup.py
__init__.py
admin.py
M
apps.py
M
models.py
tests.py
urls.py
M

Hình 28

PHẦN 3: Sử dụng hệ CSDL MongoDB:

- Xây dựng một project django riêng, sử dụng dữ liệu được xuất ra từ CSDL của hệ thống trên.
- Kiến trúc hệ thống:



Hình 29

```
from pymongo import MongoClient
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["Cars"]
mycol = mydb["cardb2"]
```

Hình 30: Xác định kết nối tới CSDL của MongoDB trong utils.py

```python
from django.db import models
from mongoengine import Document, StringField, IntField

class Car(models.Model):
    carid = models.CharField(max_length=50,null=True)
    carname = models.CharField(max_length=50,null=True)
    year = models.IntegerField(default=0)
    desintext = models.CharField(max_length=50,null=True)
    model = models.CharField(max_length=50,null=True)
    brandname = models.CharField(max_length=50,null=True)
    categoryname = models.CharField(max_length=50,null=True)
    front = models.ImageField(upload_to='uploads/fronts/')
```

Hình 31: Khai báo model trong models.py

```python
from django.http import HttpResponse
from django.shortcuts import render
from .models import Car
from utils import *

def index(request):
    return HttpResponse("Hello, this is the home page!")

def list_cars(request):

    c1 = mycol.find()
    carslist = []
    for car in c1:
        carslist.append(Car(carid=str(car['_id']),carname=car['carname'],year=car['year'],desintext=car['desintext'],model=car['model'],brandname=car['brandname'],categoryname=car['categoryname'],front=car['front']))
    context = {'carslist': carslist}
    return render(request,'list.html',context)
```

Hình 32: Lấy dữ liệu từ CSDL và tạo một array chứa dữ liệu để có thể hiển thị ra list.html